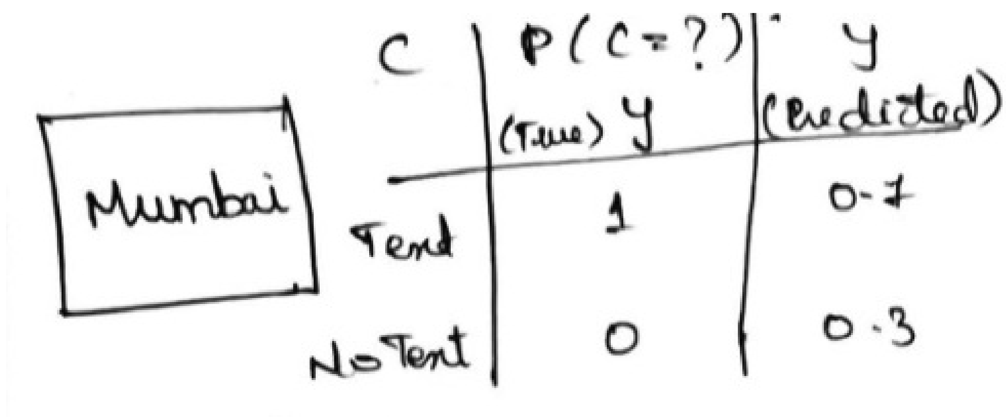
Information Theory and Gradient Descent of Sigmoid Neuron

medium.com/@manveetdn/notes-on-information-theory-and-gradient-descent-part-of-sigmoid-neuron-padhai-onefourthlabs-8e2f423a62b4

> ***Disclaimer: This is notes on "*Information Theory and Gradient Descent Part of Sigmoid Neuron*" Lesson (PadhAI* onefourthlabs *course "A First Course on Deep Learning")***

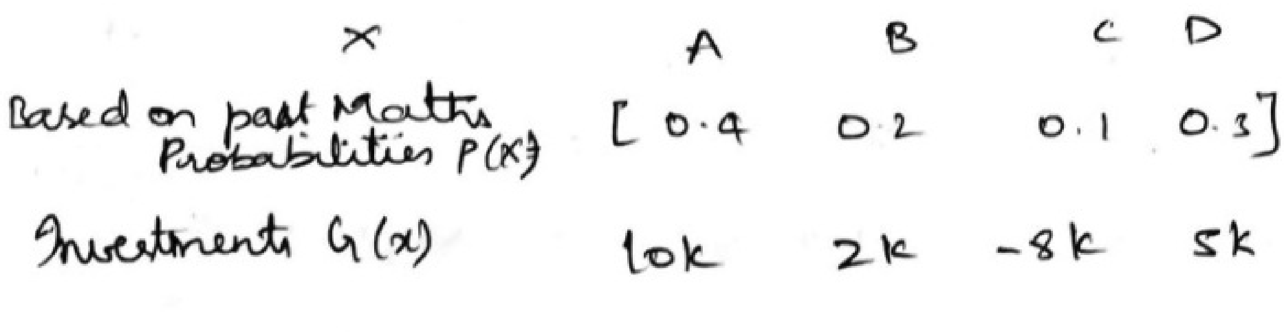


Text or noText

This is about the Text or noText classification about the true and predicted values.

Expectation:

Lets again, come to the example where we have 4 teams A,B,C,D.



Here, the expected profit is calculated as

(0.4)*(10k)+(0.2)*(2k)+(0.1)*(-8k)+(0.3)*(5k)

$$\underset{x}{E}[P] := \sum_{i \in \{A,B,C,D\}} P(X=i)\, G(X=i)$$

Formula for Expected profit.

Information Content:

Let us assume **X is a random variable of sun raises in the east**.Here, X is a sure event event it has 4 directions also.

and let **Y be a random variable** of a day with **Storm and noStorm.**

$$\underbrace{IC(X=S)}_{\text{Information content of storm i.e. } X = \text{Storm}} \propto \text{suprise} \propto \frac{1}{P(X=S)}$$

$P(X=S) \rightarrow$ probability of X=Storm

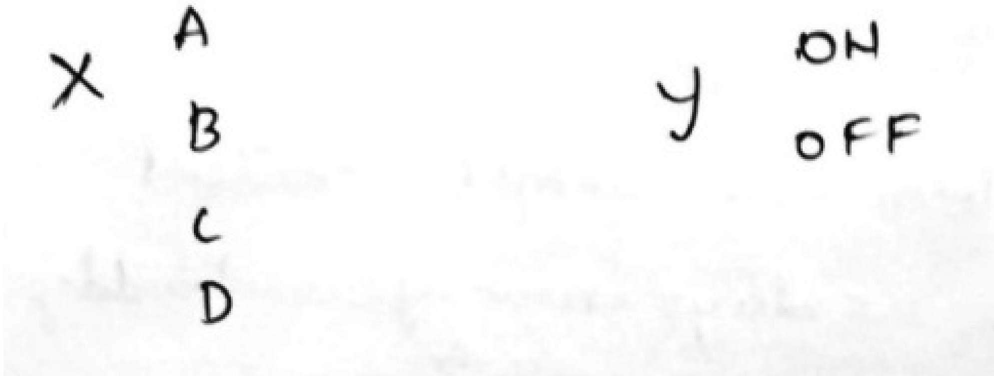
Relation of information content.

Actually **Y = Storm** is a **weighted content because there 365 days in an year hardly very few days will have a storm**. Storm is a rare event and if we tell about that information gained is very high.

Actually, until now Information content is function of the probabilities of the event **IC[P(X=S)]**.

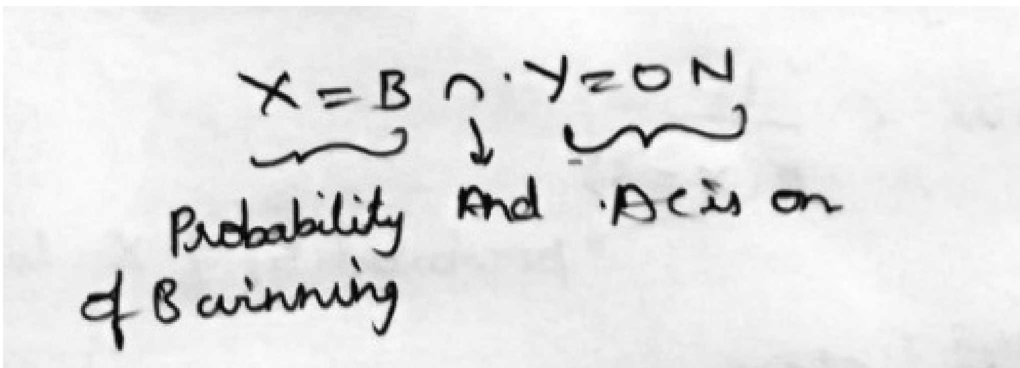
Lets again take some other example some other situation.

There is a Cricket match happening and **X is the Random variable** linked to the probabilities of **winning of team A,B,C,D playing the cricket match**.



Random variables X and Y

Y is a random variable which tells us about the **AC in the room is ON or OFF.**



The above case is about two independent events, like B team winning and AC sos on Condition the Information content of this is

IC(X=B ∩ Y = ON) this sum of individual information contents.

IC(X=B ∩ Y = ON) = IC(X=B)+IC(Y= ON)

As we know by our first intuition that Information content is function of the probabilities from above equation we can write it as

IC[P(X ∩ Y)] = IC[P(X)] +IC[P(Y)]

IC[P(X).P(Y)] = IC[P(X))]+ IC[P(Y)]

Which is of the form, **f(a.b) = f(a) + f(b).**

Only **logarithm family of curves satisfy this form/type of function** like **log(a.b) = log(a)+log(b).**

IC α 1/P(X=A) , fro that IC = log(1/P(X=A))

(1/P(X=A)) comes because more surprise less the information content.

log() is used to suffice f(a.b) = f(a) + f(b)and only log stisfie sthat condition.

$$\therefore I C(X=A) = \log\left(\frac{1}{P(X=A)}\right)$$

$$= \log 1 - \log P(X=A)$$

$$= -\log P(X=A)$$

$$\therefore \boxed{I C(X=A) = -\log_2 P(X=A)}$$

log to the base 2 is used as noraml practise. Now, we have formula for the Information content.

Form this concept we build up entropy and number of bits required to trasfer a message.

X	$P(X=?)$	$I C(X=?)$
A	$P(X=A)$	$-\log_2 P(X=A)$
B	$P(X=B)$	$-\log_2 P(X=B)$
C	$P(X=C)$	$-\log_2 P(X=C)$
D	$P(X=D)$	$-\log_2 P(X=D)$

The above table summarises all that we have learned until now and also expectation is given as

$$E[\text{Gain}] = \sum_{i=\{A,B,C,D\}} P(X=i)\,\text{Gain}(X=i)$$

Expectation is given like this.

Entropy of the random variable:

Entropy of random variable:-

$$H(x) = -\sum_{i \in A,B,C,D} P(x=i)\log_2 P(x=i)$$

It is the expected information content of a Random variable

lets take $P(x=i)$ as P_i

$$\therefore \quad H(x) = -\sum_{i \in \{A,B,C \ldots\}} P_i \log P_i$$

Relation of number of bits:

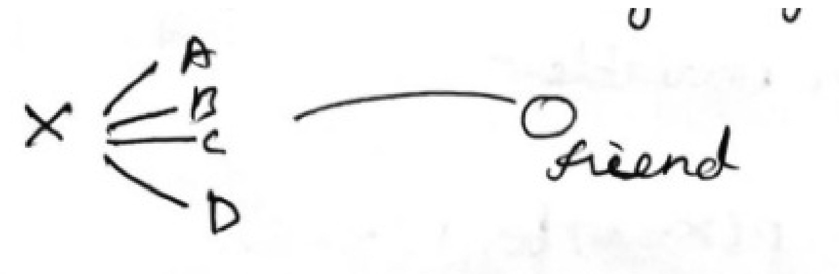
We know

$$H(x) = \sum_{i \in \{A,B,C \ldots\}} P_i \log P_i \quad \text{where } P_i = P(x=i)$$

Lets expand a bit more on this and transmit the no of bits required to transmit a message.

Suppose there is a message X we need to transmit and it can contain 4 values A, B,C, D.

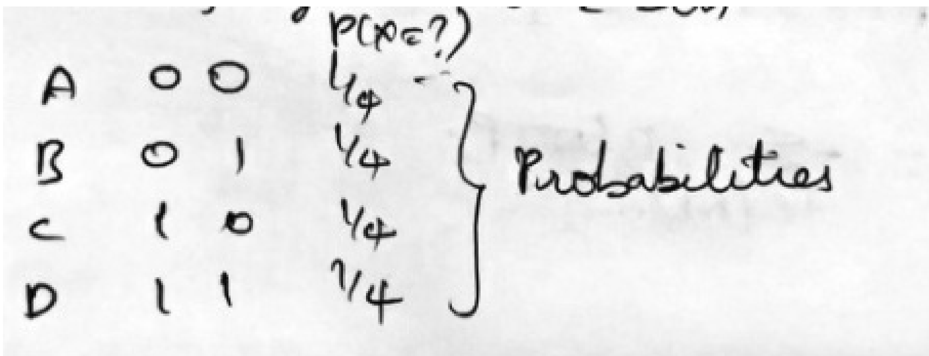
In this 4 command you are going transmit one.



Transmit a message.

According to Digital logic we need 2 bit to transmit 4 messages 0 0-A, 0 1-B,1 0-C, 1 1-D.

Here, for every message transmitting you need 2 bits.



			P(x=?)
A	0	0	1/4
B	0	1	1/4
C	1	0	1/4
D	1	1	1/4

Probabilities

Information content of each is given as

$$-\log_2 P(x=?) = -\log_2 1/4 = -\log_2 2^{-2} = 2$$

		$P(x=?)$	$-\log_2 P(x=?)$
A	0 0	1/4	2
B	0 1	1/4	2
C	1 0	1/4	2
D	1 1	1/4	2

∴ from this we can say that no. of bits required is equal to the Information content of the message

Lets take other case where message are A B C D E F G H

Now we will use 3 bits for this

		$P(x=?)$	$-\log_2 P(x=?)$
A	000	1/8	3
B	001	1/8	3
C	010	1/8	3
D	011	1/8	3
E	100	1/8	3
F	101	1/8	3
G	110	1/8	3
H	111	1/8	3

∴ Here also the number of bit required to transmit a message is equal to the Information content of the message

When you wanna send messages very frequently and the probability of sending each ,message varies then.

	Equally likely	diff prob	$-\log_2 P$
A	1/4	1/2	1
B	1/4	1/4	2
C	1/4	1/8	3
D	1/4	1/8	3

frequently sending messages
where probability send A is high
and remaining also differ

$$\begin{array}{ccccc} A & B & C & D & \\ \left[\frac{1}{2}\right. & \frac{1}{4} & \frac{1}{8} & \left.\frac{1}{8}\right] & P(x=?) \\ [1 & 2 & 3 & 3] & IC \end{array}$$

Avg bits $= \frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{8}(3) + \frac{1}{8} \times 3$

Avg bits used $= 1.75$

$\therefore$ We are going to use less number of bits than the bit occupied

KL Divergence and Cross Entropy:

$$\begin{array}{cccccc} & A & B & C & D & \\ x & [\, y_1 & y_2 & y_3 & y_4 \,] & \end{array}$$

$$I(x)\,[-\log_2 y_1 \;\; -\log_2 y_2 \;\; -\log_2 y_3 \;\; -\log_2 y_4]$$

$-\Sigma y_i \log y_i$ $\quad i = A, B, C, D$ and y is probability

True Entropy

In the above case the true entropy is given by $(-\Sigma y_i \log(y_i))$ i = A,B,C,D and y is probability.

Lets consider the n balls earn and when you asked to predict A,B,C,D by peeping into it once and **ŷ** is the one you predict when y is actual value then

$$\begin{array}{cccccc} & A & B & C & D & \\ \hat{y} & [\, \hat{y}_1 & \hat{y}_2 & \hat{y}_3 & \hat{y}_4 \,] & \cdot\, \hat{y} \end{array}$$

$$\hat{I}C(x)\,[-\log_2 \hat{y}_1 \;\; -\log_2 \hat{y}_2 \;\; -\log_2 \hat{y}_3 \;\; -\log_2 \hat{y}_3]$$

Now the actual or expected number of bits you are using will be as below.

Now the actual or expected number of bits you are using will be $-\Sigma y_i \log \hat{y}_i$ and not $-\Sigma \hat{y}_i \log \hat{y}_i$

$$-\Sigma y_i \log \hat{y}_i$$

value associated with each message is $\log_2 \hat{y}_i$

This is the probability with which we are going to get them

because that what we estimated

Now from the above concept we come into

KL Divergence:

If you know the **true distribution we estimate the average as -Σyi log(yi)**

As we don't know the **true distribution as we estimated it average** is **-Σyi log(yihat)**

Here average means the average number of bits required

$$-\sum y_i \log \hat{y}_i \qquad\qquad \sum y_i \log y_i$$

$$H_{y,\hat{y}} \text{ (Cross Entropy)} \qquad\qquad H_y \text{ (Entropy)}$$

Cross entropy and Entropy.

Now we will find the difference between the two distributions.

distance between y and yhat and we can use the difference of bits used in the **cross entropy case and entropy case**.

$$\underbrace{y \| \hat{y}}_{\text{Distance between both}} = -\sum y_i \log \hat{y}_i + \sum y_i \log y_i$$

And that is called KL Divergence

$$KL(y \| \hat{y}) = -\overbrace{\sum y_i \log \hat{y}_i}^{\text{Cross Entropy}} + \overbrace{\sum y_i \log y_i}^{\text{Entropy}}$$

computing difference b/w two distributions

Putting all together:

$$\begin{array}{c} X \\ \boxed{\text{MUMBAI}} \end{array} \quad \begin{array}{l} [1 \quad 0]\ y \\ [0.7 \quad 0.3]\ \hat{y} \end{array} \qquad \hat{y} = \frac{1}{1+e^{-(wx+b)}}$$

Square Error loss, $\text{Loss} = \sum_i (y_i - \hat{y}_i)^2$

$$KLD(y||\hat{y}) = -\sum y_i \log \hat{y}_i + \sum y_i \log y_i$$

We need to minimize the above expression w.r.t w, b

$$\min KLD(y||\hat{y}) = \min_{w,b} \left[-\sum \log y_i \log \hat{y}_i + \sum y_i \log y_i \right]$$

When we want to make minimisation with respect to the w,b then,

then Cross Entropy loss $-\sum y_i \log \hat{y}_i$

is a function of w, b and

the normal Entropy $\sum y_i \log y_i$ is a constant

$$\min_{(w,b)} f(w,b) + c$$

$$= \min_{(w,b)} f(w,b)$$

$\therefore$ minimization of whole expression minimize finally as the cross Entropy

$$KLD = \min_{(w,b)} - \sum_{i \in (\text{Text}, \text{NoText})} y_i \log \hat{y}_i$$

Now we can take a sample data set as an example and solve that using the **KL-Divergence.**

$[1 \quad 0] \; y$

$[0.7 \quad 0.3] \; \hat{y}$

$$\therefore KLD = -1 \log 0.7 - 0 \log 0.3$$

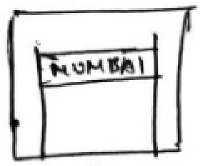
$$KLD = -\log 0.7 = -\log \hat{y}_0$$

0 — No Text, 1 — Text

Now ~~now~~ onward we are going used cross entropy loss function insted of square error loss

Now we are going proceed with two images one having text and another not having text.

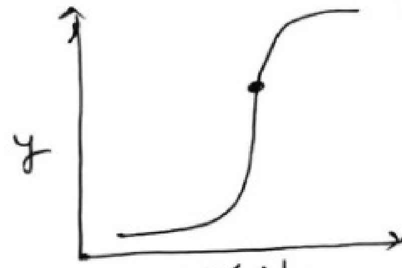
4



$x =$ image

$y = [0, 1]$

$$\hat{y} = \frac{1}{1 + e^{-(w \cdot x + b)}}$$



$\hat{y} = 0.7$

$\tilde{y} = [0.3, 0.7)$

$$L(\theta) = -\sum_{i \in \{0,1\}} y_i \log \tilde{y}_i$$

$$L(\theta) = -(y_0 \log \tilde{y}_0 + y_1 \log \tilde{y}_1)$$

$$L(\theta) = -(y_0 \log(1 - \tilde{y}_1) + y_1 \log \tilde{y}_1)$$

Text

$y = [0, 1]$

$\hat{y} = 0.7$

$\hat{y} = [0.3, 0.7]$

$$L(\theta) = -(y_0 \log(1 - \tilde{y}_1) + y_1 \log \tilde{y}_1)$$

$$L(\theta) = -(0 * \log 0.3 + 1 * \log 0.7)$$

$$= -\log 0.7 = -\log \hat{y}$$

here $\hat{y} = 0.7$ & $L(\theta) = -\log \hat{y}$

No Text

$y = [1, 0]$

$\hat{y} = 0.2$

$\tilde{y} = [0.8, 0.2]$

$$L(\theta) = -(y_0 \log(1 - \tilde{y}_1) + y_1 \log \tilde{y}_1)$$

$$L(\theta) = -(1 * \log 0.8 + 0 * \log 0.2)$$

$$= -\log 0.8 = -\log(1 - \hat{y})$$

here $\hat{y} = 0.2$ & $L(\theta) = -\log(1 - \hat{y})$

$\therefore$ loss function is given as

$$L(\theta) = -((1-y)\log(1-\hat{y}) + y\log\hat{y})$$

When true output is 1 $(y=1)$

$L(\theta) = -\log\hat{y}$ which is same as before one

when true output is 0 $(y=0)$

$L(\theta) = -\log(1-\hat{y})$ which is same as before

$$\therefore L(\theta) = -((1-y)\log(1-\hat{y}) + y\log\hat{y})$$

Learning Algorithm :-

Initialize w, b

Iterate over data

compute $\hat{y}$

compute $L(w,b)$

$w_{t+1} = w_t - \eta \Delta w_t$

$b_{t+1} = b_t - \eta \Delta b_t$

till satisfied

Input	output
1.2	0
-2.1	1
3.2	0
-0.5	1
0.6	1

$$\hat{y} = \frac{1}{1+e^{-(wx+b)}}$$

$$L(\theta) = -\sum_{i=1}^{5}((1-y_i)\log(1-\hat{y}_i) + y_i\log\hat{y}_i)$$

$$L(\theta) = -[(1-y)\log(1-\hat{y}) + y\log\hat{y}]$$

We need to compute Δw. $\Delta w = \frac{\partial L(\theta)}{\partial w} = \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$

$$\frac{\partial L(\theta)}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\{-(1-y)\log(1-\hat{y}) - y\log\hat{y}\}$$

$$= (-1)(-1)\frac{(1-y)}{(1-\hat{y})} - \frac{y}{\hat{y}}$$

$$= \frac{\hat{y}(1-y) - y(1-\hat{y})}{(1-\hat{y})\hat{y}}$$

$$= \frac{\hat{y} - y}{(1-\hat{y})\hat{y}}$$

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= -\frac{1}{(1+e^{-(wx+b)})^2}\frac{\partial}{\partial w}(e^{-(wx+b)})$$

$$= \frac{1}{(1+e^{-(wx+b)})^2} * (e^{-(wx+b)})\frac{\partial}{\partial w}(-(wx+b))$$

$$= \frac{1}{(1+e^{-(wx+b)})} * \left(\frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})}\right) * (-x)$$

$$= \hat{y} * (1-\hat{y}) * x$$

$$\Delta w = \frac{\partial L(\theta)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w} = \frac{\hat{y} - y}{\cancel{(1-\hat{y})}\cancel{\hat{y}}} * \cancel{\hat{y}} * \cancel{(1-\hat{y})} * x$$

$$\Delta w = (\hat{y} - y) * x$$

like that we will compute Δw and place in the algorithm

```
X = [0.5, 2.5]
Y = [0.2, 0.9]

def f(w, b, x):
   #sigmoid with parameters w, b
   return 1.0 / (1.0 + np.exp(-(w*x + b))

def error(w, b):
     err = 0.0
     for x, y in zip(X, Y):
        fx = f(w, b, x)
        err += 0.5* (fx - y) ** 2
     return err

def grad_b(w, b, x, y):
    fx = f(w, b, x)
    return (fx - y) * fx * (1 - fx)

def grad_w(w, b, x, y):
    fx = f(w, b, x)
    return (fx - y) * fx * (1 - fx) * x


def do_gradient_descent():
    w, b, eta = -2, -2, 1.0
    max_epochs = 1000
    for i in range(max_epochs):
        dw, db = 0, 0
        for x, y in zip(X, Y) :
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        w = w - eta * dw
        b = b - eta * db
```

The code of sigmoid neuron before using cross entropy.

```
def grad_w(w, b, x, y):
    fx = f(w, b, x)
    return (fx - y) * x
```

```
def grad_b(w, b, x, y):
    fx = f(w, b, x)
    return (fx - y)
```

```python
def error(w, b):
     err = 0.0
     for x, y in zip(X, Y):
        fx = f(w, b, x)
        err +=  - [(1 - y) * math.log (1 - fx , 2) + y * math.log (fx, 2)]
     return err
```

Changes that need to made in the error, grad_w,grad_b function.

Small changes in the code need to be done for thus cross entropy loss error funtion , grab_b, grad_w changes will take place

This is all about using gradient descent and Cross Entropy loss for the Sigmoid Neuron (5)

```
def error(w,b):
    err = 0
    for x,y in zip(X,Y):
        fx = f(w,b,x)
        err += - [(1-y)* math.log(1-fx,2) + y* math
                                        .log(fx,2)]
    return err
```

This is the cross entropy loss funtion

```
def grad_b(w,b,x,y):
    fx = f(w,b,x).
    return (fx-y)
```

This formula which we derived for Δb

```
def grad_w(w,b,x,y):
    fx = f(w,b,x)
    return (fx-y)*x
```

This is the formula for the Δw which also we derived befor $\Delta w = (\hat{y} - y) * x$

This is a small try, uploading the notes . I believe in “**Sharing knowledge is that best way of developing skills**”.Comments will be appreciated. Even small edits can be suggested.

> **Each Applause will be a great encouragement.**

Do follow my medium for more updates……